Abgabe DSR
Testat 2

01.12.2023

Gruppe 2
Nicole Enzelsberger
Michelle Göppinger
Maximilian Hehl

Zentrale Grenzwertsatz

Zentrale Grenzwertsatz

Für jedes $n = 1, 2, \ldots$ seien die Zufallsvariablen $X_1, X_2, \ldots, X_n$ unabhängig und besitzen die gleiche Verteilung mit dem Erwartungswert $\mu = E(X_i)$ und der Varianz $\sigma^2 = Var(X_i) =$. Dann gilt für die Verteilungsfunktion der standardisierten Summen $G_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$:

$$\lim_{n\to\infty} P\left(G_n \leq x\right) = \Phi(x)$$

Dabei ist Φ die Verteilungsfunktion der Standard-Normalverteilung $N(0, 1)$.

Überprüfen Sie mit R den zentralen Grenzwertsatz.

Verteilung

Erzeugen Sie einen Data Frame, der die gezogenen Zufallszahlen G_n enthält. Die Stichprobe soll einen Umfang von 50000 haben (Anzahl Zeilen der Tabelle). Der Data Frame hat sechs Spalten, die die Werte für die Zahlen $n = 1, 2, 3, 10, 100, 1000$ enthalten. Geben Sie die ersten 10 Zeilen des Data Frames aus.

Welche Verteilung Sie für X_i nehmen, bleibt Ihnen überlassen. Es soll nur keine Normalverteilung sein.

```
options(warn=-1)
# G_n: Umfang der Stichprobe
# n: Anzahl der Zufallszahlen

n_rows = 50000

g = data.frame(
    G_1 = rt(n_rows, 1),
    G_2 = rt(n_rows, 2),
    G_3 = rt(n_rows, 3),
    G_10 = rt(n_rows, 10),
    G_100 = rt(n_rows, 100),
    G_1000 = rt(n_rows, 1000)
)
print(dim(g))
```

```
## [1] 50000     6
```

```
print(head(g))
```

```
##           G_1         G_2        G_3       G_10       G_100     G_1000
## 1 -0.50651247 -0.92572105  1.3873983  0.4174636 -0.04092938  0.4944634
## 2 -2.56369497 -0.67087480  0.1376392  1.9257948  0.38167903  0.3079259
```

```
## 3  1.72606574  0.87644179 -0.3610023  0.1146069 -0.85105073  1.1895846
## 4 22.28851337  0.04072003  1.1351572 -1.8207777  0.50783374 -0.4173495
## 5  0.08645695 -0.42747464  1.4753047 -2.0641964  0.01608835  1.1691136
## 6 -0.03669335 -0.13711737 -0.9079204  0.7626160 -0.28306348 -2.0547245
```

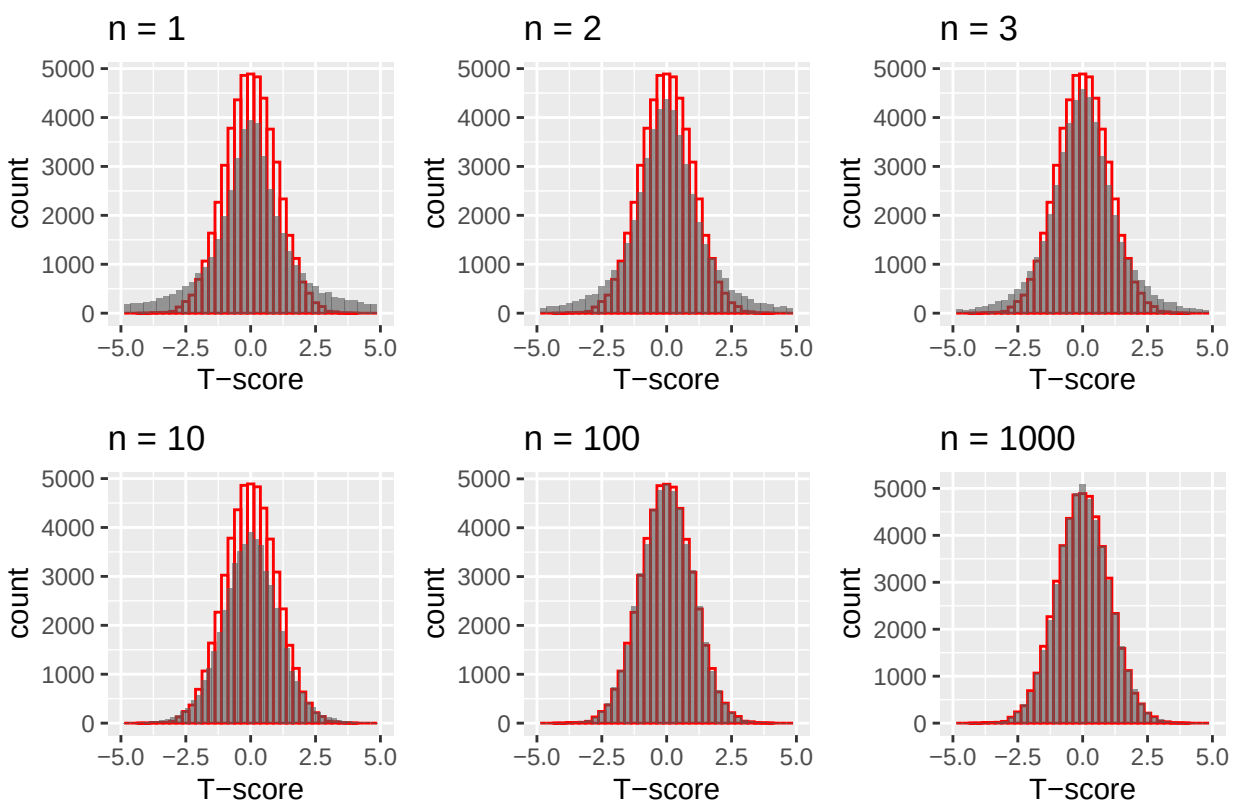
Plot

Plotten Sie sechs Histogramme, die je für $n = 1, 2, 3, 10, 100, 1000$ die Verteilung im Vergleich zu einer $(0, 1)$-Normalverteilung zeigen. Die Intervallbreite soll 0,25 sein.

Tipp: Platzieren Sie die sechs Plots auf einem 3×2-Gitter.

```
library(ggplot2)
library(gridExtra)
library(grid)

normal_dist = data.frame(normal = rnorm(n_rows, 0, 1)) # Normalverteilung mit 50000 Zufallszahlen
alpha_t = 0.6

p1 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_1), binwidth = 0.25, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 1", x = "T-score", y = "count")

p2 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_2), binwidth = 0.25, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 2", x = "T-score", y = "count")

p3 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_3), binwidth = 0.25, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 3", x = "T-score", y = "count")

p4 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_10), binwidth = 0.2, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 10", x = "T-score", y = "count")

p5 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_100), binwidth = 0.25, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 100", x = "T-score", y = "count")

p6 = ggplot(g) +
    geom_histogram(data = normal_dist, aes(x = normal), binwidth = 0.25, color = "red", fill="transparer
    geom_histogram(aes(x = G_1000), binwidth = 0.25, alpha = alpha_t) +
    xlim(-5, 5) +
    labs(title = "n = 1000", x = "T-score", y = "count")
```

```r
grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 3, top=textGrob("Histogramme der Standardnormalverteilung (
```



Abweichung

Nun soll die Abweichung der Standardisierten G von der Normalverteilung für alle n quantifiziert werden. Hierzu soll für alle Balken (bins) der Histogramme aus der vorigen Aufgabe die Differenz von G zur Normalverteilung gebildet und quadriert werden. Diese Werte werden aufaddiert und durch die Anzahl der Intervalle geteilt. Daraus wird die Wurzel gezogen.

Sie können sich auch ein anderes Maß zur Bestimmung der Abweichung überlegen.

Geben Sie die Abweichungen aus. Stimmt es, dass die Abweichungen mit größerem n kleiner werden?

Tipp: `hist(plot = FALSE)` erzeugt ein Histogramm, ohne es zu plotten. Gerne können Sie auch Ihr eigenes Histogramm nutzen.

```r
library(knitr)

table = data.frame(n = c(1, 2, 3, 10, 100, 1000), mean_diff = c(0, 0, 0, 0, 0, 0))
for (i in 1:6) { # vergleich von jedem n mit Normalverteilung pro Bin
    histogram = hist(g[, i], plot = FALSE, binwidth = 0.25)
    histogram_normal = hist(normal_dist$normal, plot = FALSE, binwidth = 0.25)
    sqr_diff = (histogram$counts - histogram_normal$counts)^2
    mean_diff = sqrt(sum(sqr_diff) / length(sqr_diff))
    table[i, 2] = mean_diff
```

```
}
print(kable(table))
```

```
##
##
## |    n|  mean_diff|
## |----:|----------:|
## |    1| 8166.13890|
## |    2| 8349.72817|
## |    3| 8918.95733|
## |   10| 3332.88975|
## |  100|   64.10322|
## | 1000| 1493.84398|
```

```
# Die Abweichungen werden kleiner, je größer n ist.
```

Untersuchungen zur koronaren Herzkrankheit (t-tests)

Untersuchungen zur koronaren Herzkrankheit

In diesem Abschnitt sollen Daten von Probanden bzw. Patienten auf das Risiko für koronare Herzkrankheit untersucht werden. Dies ist eine Erkrankung der Herzkranzgefäße (Koronararterien), die sich durch Ablagerungen in den Gefäßwänden verengen. Der Original-Herz-Datensatz ist unter

- https://archive.ics.uci.edu/ml/datasets/Heart+Disease

beschrieben. Wir nutzen eine konsolidierte CSV-Datei, die bereits Header enthält. Download unter:

- https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download

Die Datei enthält 13 Merkmale, die einen Einfluss auf eine koronare Herzkrankheit haben können. Das 14. Merkmal `goal` (im Original auch `num`) ist die Diagnose (Klassifizierung). Der Wert ist 0, falls keine krankhafte Verengung der Gefäße vorliegt, oder 1, 2, 3 oder 4, falls – je nach Stärke – eine krankhafte Verengung der Gefäße vorliegt. Wir unterscheiden im Folgenden nur die Zustände "gesund" (0) und "krank" (1, 2, 3, 4). Unter

- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names

finden Sie eine Beschreibung aller Attribute. Hier ist eine Zusammenfassung. Wir benötigen insbesondere die Merkmale `sex`, `trestbps`, `chol` und `goal`.

Feld	Bedeutung
age	age in years
sex	sex (1 = male; 0 = female)
cp	chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)
trestbps	resting systolic blood pressure (in mmHg on admission to the hospital)
chol	serum cholestoral in mg/dl
fbs	fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
restecg	resting electrocardiographic results (0 = normal; 1 = having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV); 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
thalach	maximum heart rate achieved
exang	exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest.[1]
slope	slope of the peak exercise ST segment (1 = upsloping; 2 = flat; 3 = downsloping)
ca	number of major vessels (0-3) colored by flourosopy
thal	3 = normal; 6 = fixed defect; 7 = reversable defect

[1] ST depression refers to a finding on an electrocardiogram, wherein the trace in the ST segment is abnormally low below the baseline.

Feld	Bedeutung
goal	diagnosis of heart disease (0: $< 50\%$ diameter narrowing ; 1, 2, 3, 4: $> 50\%$ diameter narrowing)

Einlesen der Herz-Daten

Lesen Sie die Datei aus der URL als Data Frame zur weiteren Bearbeitung ein. Überlegen Sie, ob sie Faktoren sinnvoll einsetzen können. Geben Sie die ersten drei Zeilen und fünf Spalten aus[2]:

```
#Einlesen der Datei
herz = read.csv(url("https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download"))

herz$sex = factor(herz$sex, levels = c(0, 1), labels = c("f", "m"))
herz$goal = factor(herz$goal, levels = c(0, 1, 2, 3, 4), labels = c("gesund", "krank", "krank", "krank"

# Ausgabe der ersten 3 Zeilen und 5 Spalten

head(herz, n = c(3, 5))
```

```
##   age sex cp trestbps chol
## 1  63   m  1      145  233
## 2  67   m  4      160  286
## 3  67   m  4      120  229
```

Cholesterin im Vergleich Männer/Frauen

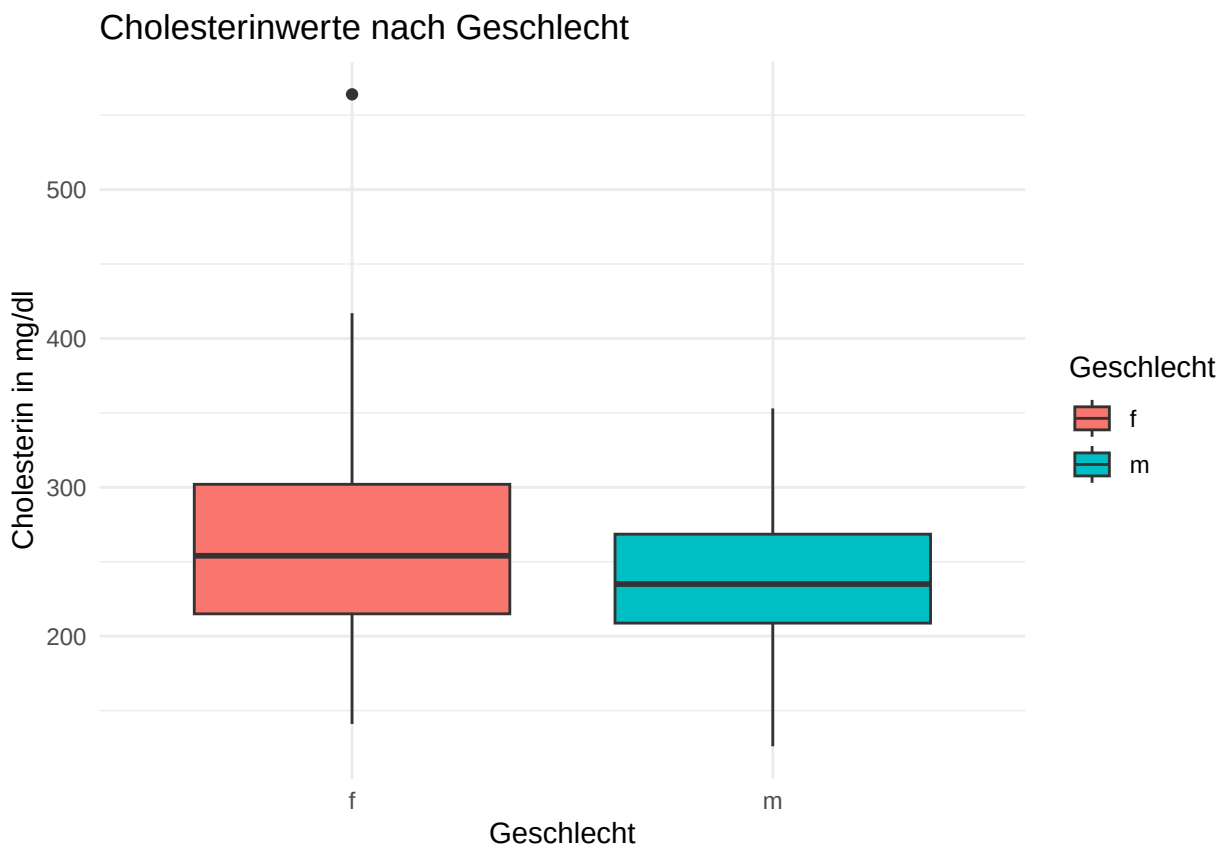
Nun sollen die Cholerstin-Werte untersucht werden – zunächst im Vergleich Männer zu Frauen.

Überblick über Cholesterin-Daten

Verschaffen Sie sich einen Überblick, indem Sie ein Boxplot für das Cholesterin gruppiert nach dem Geschlecht plotten.

```
ggplot(herz, aes(x = sex, y = chol, fill = sex)) +
  geom_boxplot() +
  labs(title = "Cholesterinwerte nach Geschlecht",
       x = "Geschlecht",
       y = "Cholesterin in mg/dl") +
  scale_fill_discrete(name = "Geschlecht") +
  theme_minimal()
```

[2] Möglicherweise kommt es zu einem Fehler beim Einlesen des ersten Attributs (`age`). Manuelles Umbenennen hilft.



Konfidenz-Intervall

Berechnen Sie das Konfidenz-Intervall (Niveau 95%) für den Cholesterin-Level jeweils für Männer und Frauen.

Tabelle Geben Sie das Ergebnis als `kable`-Tabelle aus:

```
# Filtern nach Geschlecht & berechnen des Konfidenzintervalls

konf_chol_men = t.test(herz[herz$sex == "m", ]$chol, conf.level = 0.95)$conf.int
konf_chol_women = t.test(herz[herz$sex == "f", ]$chol, conf.level = 0.95)$conf.int

# Erstellen der Tabelle
result_table_men_vs_women = data.frame(Geschlecht = c("Männer", "Frauen"),
                          Untere_Grenze = c(konf_chol_men[1], konf_chol_women[1]),
                          Obere_Grenze = c(konf_chol_men[2], konf_chol_women[2])
                          )
kable(result_table_men_vs_women)
```

Geschlecht	Untere_Grenze	Obere_Grenze
Männer	233.7432	245.4607
Frauen	248.6722	274.8330

Überlappung? Überlappen sich die Bereiche?

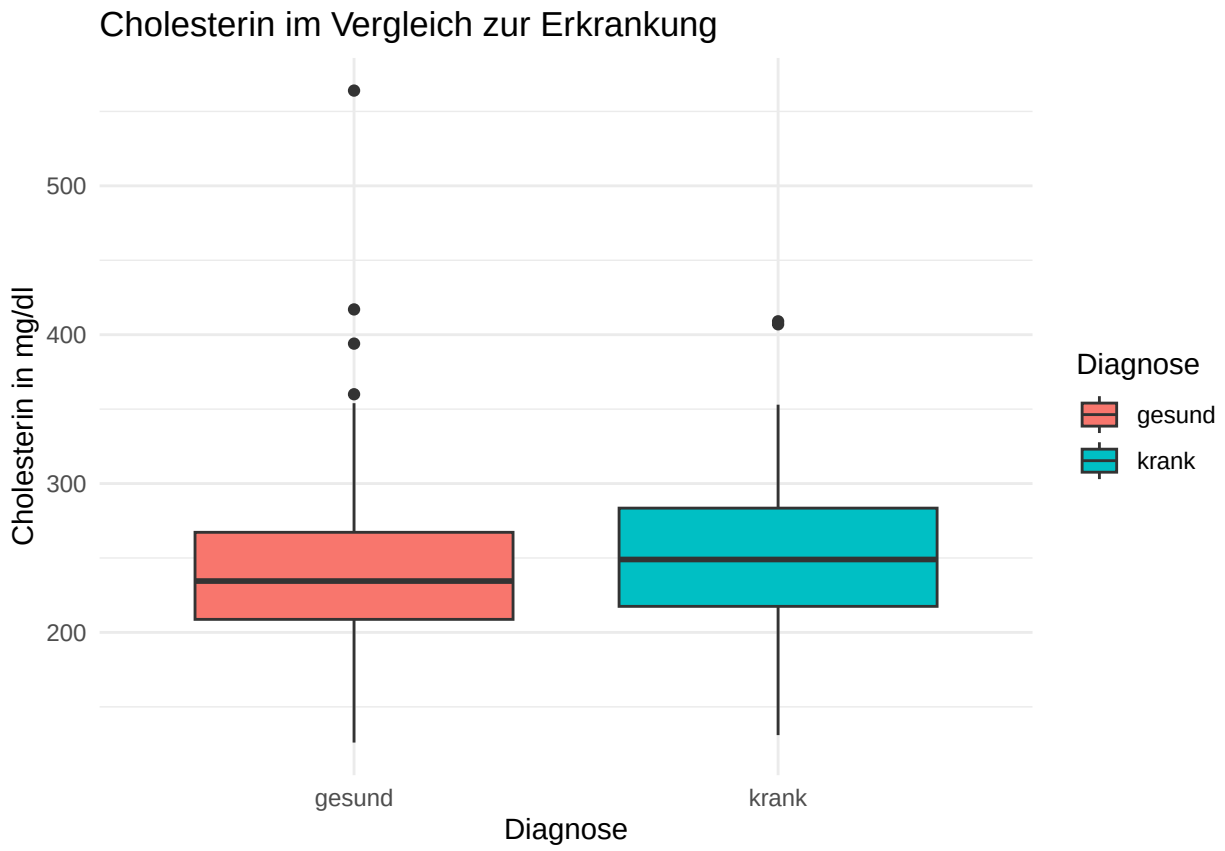
Cholesterin im Vergleich zur Erkrankung

Nun sollen die Cholesterin-Werte in Abhängigkeit der Diagnose untersucht werden.

Überblick über Cholesterin-Daten

Verschaffen Sie sich einen Überblick, indem Sie ein Boxplot für das Cholesterin gruppiert nach der Diagnose plotten.

```
ggplot(herz, aes(x = goal, y = chol, fill = goal)) +
  geom_boxplot() +
  labs(title = "Cholesterin im Vergleich zur Erkrankung",
       x = "Diagnose",
       y = "Cholesterin in mg/dl") +
  scale_fill_discrete(name = "Diagnose") +
  theme_minimal()
```



Konfidenz-Intervall

Berechnen Sie die Konfidenzintervalle für beide Gruppen und geben Sie das Ergebnis als `kable`-Tabelle aus:

```
konf_gesund = t.test(herz[herz$goal == "gesund", ]$chol, conf.level = 0.95)$conf.int
konf_krank = t.test(herz[herz$goal == "krank", ]$chol, conf.level = 0.95)$conf.int
```

```r
result_table_gesund_vs_krank = data.frame(Diagnose = c("Gesund","Krank"),
                                          Untere_Grenze = c(konf_gesund[1], konf_krank[1]),
                                          Obere_Grenze = c(konf_gesund[2], konf_krank[2])
                                          )
kable(result_table_gesund_vs_krank)
```

Diagnose	Untere_Grenze	Obere_Grenze
Gesund	234.3977	250.8828
Krank	243.1752	259.7744

```
### Test

Es sieht so aus, als ob der Cholesterin-Wert bei den erkrankten Patienten höher ist als bei den nicht e

#### Wie lauten die Hypothesen?

Formulieren Sie die Hypothesen ($H_0$ und $H_1$).


#### Testanwendung

Wenden Sie den Test mit R an. Was ist das Ergebnis?


'''r
# Nullhypothese (H0) = Cholesterinwert bei Kranken ist gleich oder kleiner wie bei Gesunden
# Alternativhypothese (H1) Cholesterinwert bei Kranken ist höher als bei Gesunden


test_diagnose = t.test(herz$chol[herz$goal == "krank"],
                       herz$chol[herz$goal == "gesund"],
                       alternative = "greater"
                       ) # betrachten das der Cholesterinwert bei Kranken höher ist als bei Gesunden

print(test_diagnose)

##
##  Welch Two Sample t-test
##
## data:  herz$chol[herz$goal == "krank"] and herz$chol[herz$goal == "gesund"]
## t = 1.4924, df = 298.64, p-value = 0.06832
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -0.9327489        Inf
## sample estimates:
## mean of x mean of y
##  251.4748  242.6402
```

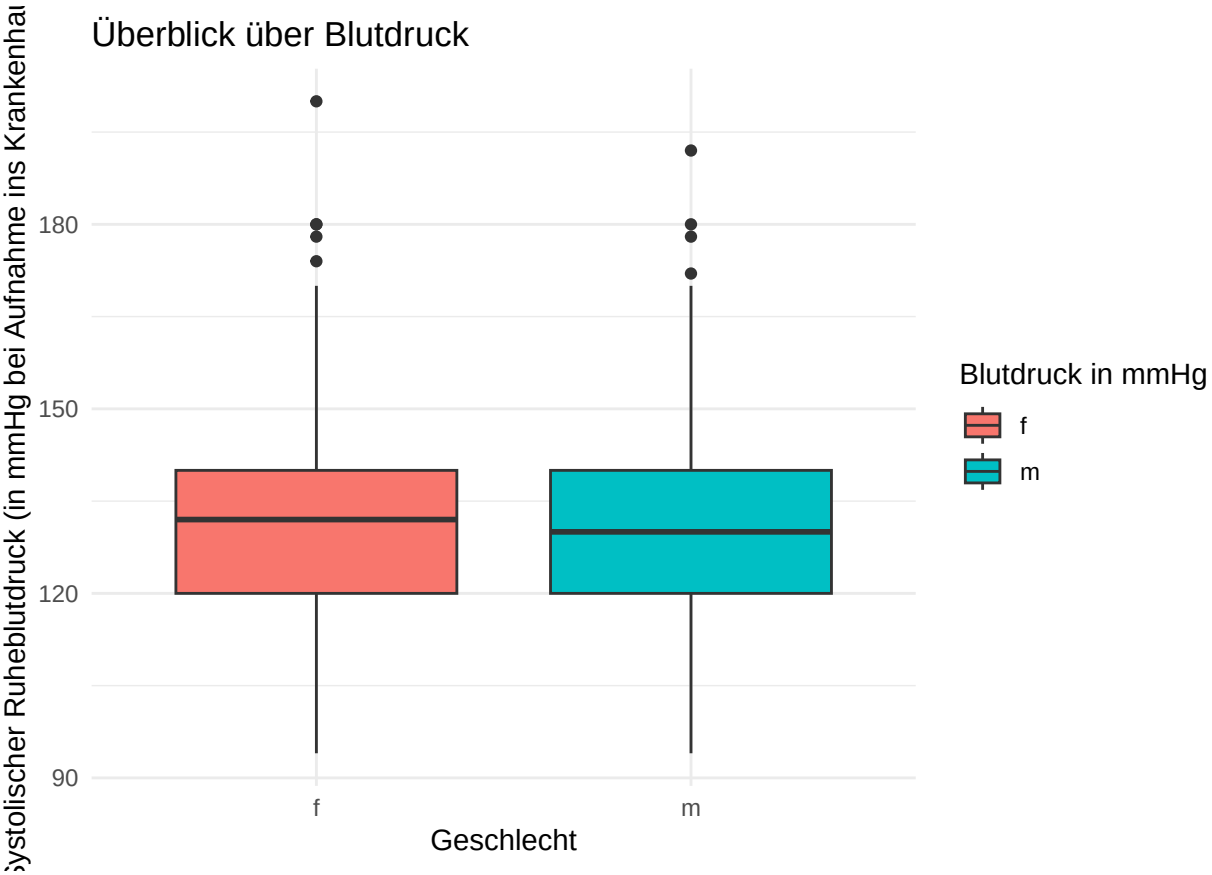
Systolischer Ruheblutdruck

Der systolische Blutdruck liegt beim gesunden Menschen bei ca. 120 mmHg.

Überblick über Blutdruck

Plot Verschaffen Sie sich einen Überblick, indem Sie ein Boxplot für den Blutdruck in Ruhe gruppiert nach dem Geschlecht plotten.

```
ggplot(herz, aes(x = sex, y = trestbps, fill = sex)) +
  geom_boxplot() +
  labs(title = "Überblick über Blutdruck",
       x = "Geschlecht",
       y = "Systolischer Ruheblutdruck (in mmHg bei Aufnahme ins Krankenhaus)") +
  scale_fill_discrete(name = "Blutdruck in mmHg") +
  theme_minimal()
```



Normalverteilt? Kann überhaupt davon ausgegangen werden, dass die Daten normalverteilt sind?

Konfidenzintervalle nach Erkrankung

Berechnen Sie die Konfidenzintervalle für den Ruheblutdruck aufgeschlüsselt nach der Diagnose (erkrankt/nicht erkrankt) und geben Sie das Ergebnis als `kable`-Tabelle aus:

```
konf_blutdruck_gesund = t.test(herz[herz$goal == "gesund", ]$trestbps, conf.level = 0.95)$conf.int
konf_blutdruck_krank = t.test(herz[herz$goal == "krank", ]$trestbps, conf.level = 0.95)$conf.int

result_table_blutdruck = data.frame(Diagnose = c("Gesund","Krank"),
```

```
                                        Untere_Grenze = c(konf_blutdruck_gesund[1], konf_blutdruck_kra
                                        Obere_Grenze = c(konf_blutdruck_gesund[2], konf_blutdruck_kran
                                        )
kable(result_table_blutdruck)
```

Diagnose	Untere_Grenze	Obere_Grenze
Gesund	126.7514	131.7486
Krank	131.4205	137.7161

Test, ob Kranke höheren Ruhe-Blutdruck haben

Überprüfen Sie mit einem Hypothesen-Test, ob Erkrankte einen höheren Ruhe-Blutdruck haben als gesunde Probanden.

Wie lauten die Hypothesen? Formulieren Sie die Hypothesen (H_0 und H_1).

Testanwendung Wenden Sie den Test mit R an. Was ist das Ergebnis?

```
# H0 Kranke haben einen niedrigeren oder gleichen Blutdruck wie Gesunde
# H1 Kranke haben einen höheren BLutdruck als Gesunde

test_blutdruck = t.test(herz$trestbps[herz$goal == "krank"],
                        herz$trestbps[herz$goal == "gesund"],
                        alternative = "greater"
                        ) # betrachten das Blutdruck bei Kranken höher ist als bei Gesunden

print(test_diagnose)
```

```
##
##  Welch Two Sample t-test
##
## data:  herz$chol[herz$goal == "krank"] and herz$chol[herz$goal == "gesund"]
## t = 1.4924, df = 298.64, p-value = 0.06832
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -0.9327489        Inf
## sample estimates:
## mean of x mean of y
##  251.4748  242.6402
```

Unüberwachtes Lernen mit dem Herz-Datensatz

Clustering und PCA auf die Herzdaten

Einlesen der Herz-Daten

Es werden wieder die Herzdaten aus der letzten Aufgabe genutzt. Lesen Sie diese als Data Frame ein.

```
df = read.csv(url('https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download'))
```

Bedeutet "ähnliche Merkmale" auch "gleiche Diagnose"?

Für jeden Datensatz ist bekannt, zu welcher Klasse er gehört: 0 (gesund) und 1 (erkrankt). Wir wollen untersuchen, wie gut *ähnliche* Datensätze zur gleichen Klasse gehören. Dafür soll mit dem k-means-Clusterverfahren der Datensatz in zwei Cluster eingeteilt werden.

Nur reelle Merkmale

Zunächst sollen **nur die nummerischen Merkmale** benutzt werden und nicht jene, die Faktoren sind.

Clustering Clustern Sie diese Daten. Überlegen Sie, ob Sie die Daten standardisieren wollen.

```
print(head(df[sapply(df, is.numeric)]))
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope goal
## 1  63   1  1      145  233   1       2     150     0     2.3     3    0
## 2  67   1  4      160  286   0       2     108     1     1.5     2    2
## 3  67   1  4      120  229   0       2     129     1     2.6     2    1
## 4  37   1  3      130  250   0       0     187     0     3.5     3    0
## 5  41   0  2      130  204   0       2     172     0     1.4     1    0
## 6  56   1  2      120  236   0       0     178     0     0.8     1    0
```

```
# numerische Merkmale: age, trestbps, chol, thalach, oldpeak, ca
# faktorielle Merkmale: sex, cp, fbs, restecg, exang, slope, thal, goal
# Merkmale initial nicht richtig zugeordnet -> umwandeln
conv_to_factor = c("sex", "cp", "fbs", "restecg", "exang", "slope", "goal")
df_clust = df %>% mutate_at(conv_to_factor, factor)
df_clust$ca = as.numeric(as.factor(df_clust$ca))

# add df for healthy/sick (1= healtyh, 0=sick)
healthy = df_clust$goal!=0

# select numerical values
```

```r
df_nums = df_clust %>% select_if(is.numeric)

# scale data (features have different scales
# -> features have different influence on distance calculation)
df_nums = data.frame(scale(df_nums))

# clustering, nstart is the number of random initial cluster centers
set.seed(42)
km_res = kmeans(df_nums, centers=2, nstart=10)
```

Richtig? Berechnen Sie, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus.

Hinweis: Berücksichtigen Sie, dass die Vergabe der Clusternummern zufällig ist. D.h. sowohl die Cluster (1, 2) wie auch (2, 1) sind möglich.

```r
cluster1 = sapply(km_res$cluster, function(X) X==1)
cluster2 = sapply(km_res$cluster, function(X) X==2)

perc_cluster1 = round(mean(cluster1 == healthy) * 100, digits = 2)
perc_cluster2 = round(mean(cluster2 == healthy) * 100, digits = 2)

# clusters have no semantic meaning (healthy/sick) -> choose cluster with higher correspondence
if (perc_cluster1 > perc_cluster2) {
    km_healthy = cluster1
    max_perc = perc_cluster1
} else {
    km_healthy = cluster2
    max_perc = perc_cluster2
}

print(paste(max_perc, "% der Datensätze wurden richtig einem Cluster zugeordnet"))
```

```
## [1] "74.59 % der Datensätze wurden richtig einem Cluster zugeordnet"
```

Scatterplot age vs. thalach Plotten Sie die Merkmale `age` und `thalach` als Scatterplot. Färben Sie die Punkte gemäß ihrer Clusterzuordnung ein. Die Form (`shape`) eines Punkts soll zeigen, ob die Klassifikation (d.h. der Cluster) richtig oder falsch ist.

```r
df_nums$cluster = as.factor(km_res$cluster)
df_nums$correct = km_healthy == healthy

ggplot(df_nums) +
    geom_point(aes(x=age, y=thalach, color=cluster, shape=correct), size=3)
```

Mit Dummy-Variablen

Nun sollen **alle Merkmale** benutzt werden.

Clustering Clustern Sie diese Daten. Überlegen Sie, wie die Faktoren zu Zahlen werden.

```
# convert factors to numeric
df_dummy = df %>% mutate_if(function(x) !is.numeric(x), function(y) as.numeric(as.factor(y)))

# scale data
df_dummy = data.frame(scale(df_dummy))

# clustering
km_res = kmeans(df_dummy, centers=2, nstart=10)
```

Richtig? Berechnen Sie für diesen Fall, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus. Wie hat sich der Wert verändert? Warum ist dies so?

```
cluster1 = sapply(km_res$cluster, function(X) X==1)
cluster2 = sapply(km_res$cluster, function(X) X==2)

perc_cluster1 = round(mean(cluster1 == healthy) * 100, digits = 2)
```

```
perc_cluster2 = round(mean(cluster2 == healthy) * 100, digits = 2)

if (perc_cluster1 > perc_cluster2) {
    km_healthy = cluster1
    max_perc = perc_cluster1
} else {
    km_healthy = cluster2
    max_perc = perc_cluster2
}

print(paste(max_perc, "% der Datensätze wurden richtig einem Cluster zugeordnet"))
```

```
## [1] "85.15 % der Datensätze wurden richtig einem Cluster zugeordnet"
```

Scatterplot age vs. thalach Plotten Sie erneut und schauen Sie, wie die richtigen nun Punkte verteilt sind.

```
df_dummy$cluster = as.factor(km_res$cluster)
df_dummy$correct = km_healthy == healthy

ggplot(df_dummy) +
    geom_point(aes(x=age, y=thalach, color=cluster, shape=correct), size=3) +
    scale_color_discrete(drop = FALSE)
```

PCA

Wenden Sie eine PCA auf diesen Datensatz an. Es sollen alle Merkmale berücksichtigt werden.

Wichtige Merkmale

Welche Merkmale der ersten Hauptkomponente tragen am meisten zur Varianz bei? Geben Sie die TOP-10-Merkmale an.

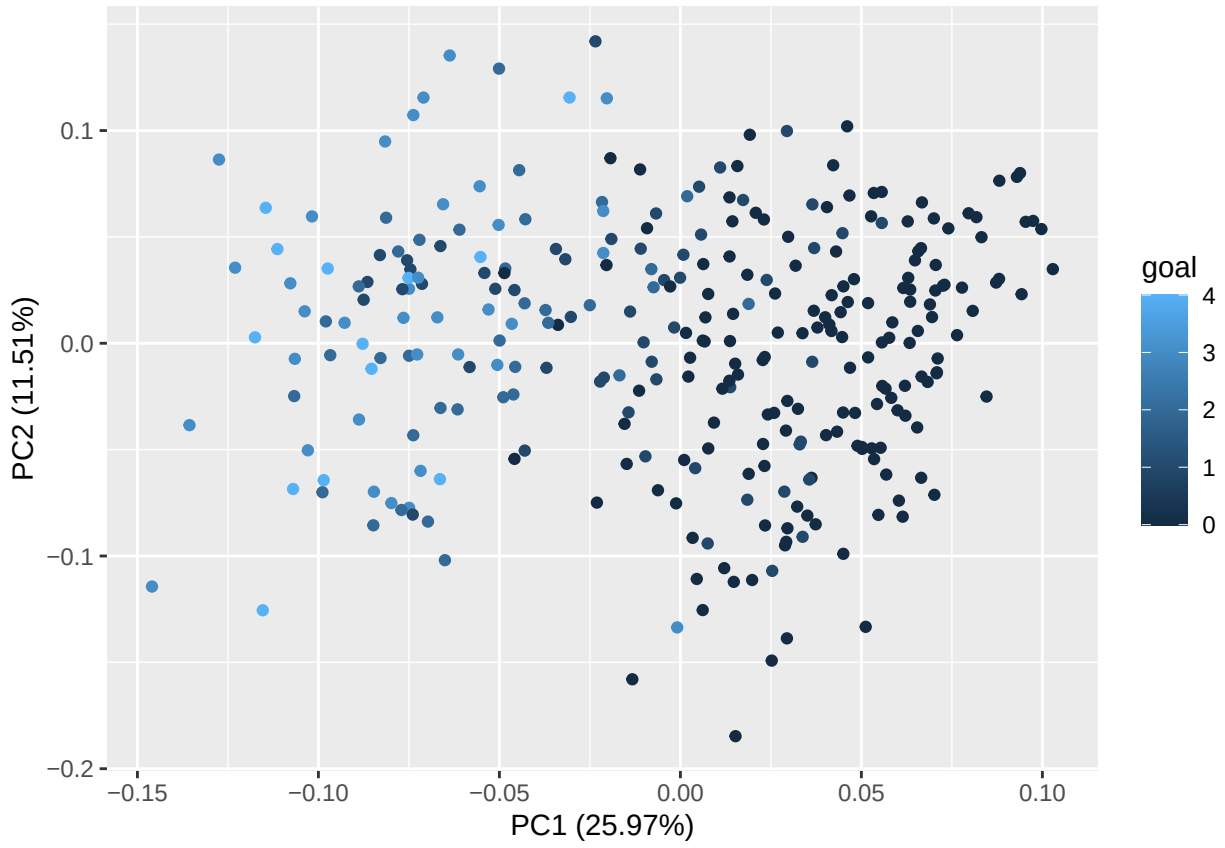
```
# pca only works with numerical values -> all features should be considered
# -> convert to numeric values
df_pca = df %>% mutate_if(function(x) !is.numeric(x), function(y) as.numeric(as.factor(y)))

# apply pca
pca_res = prcomp(df_pca, scale=TRUE)

# positive values: positive correlation
# negative values: negative correlation
pc1 = pca_res$rotation
first_comp = data.frame(pca_res$rotation[, 1])

# the higher the absolute value, the more it contributes to the variance
first_comp_desc = abs(first_comp) %>% arrange(desc(pca_res.rotation...1.))
head(first_comp_desc,10)
```

```
##          pca_res.rotation...1.
## goal                 0.4253600
## oldpeak              0.3633676
## thalach              0.3426867
## thal                 0.3172153
## slope                0.3113089
## exang                0.3021607
## ca                   0.2994957
## cp                   0.2686758
## age                  0.2397569
## trestbps             0.1398275
```

Erste und zweite Hauptkomponente

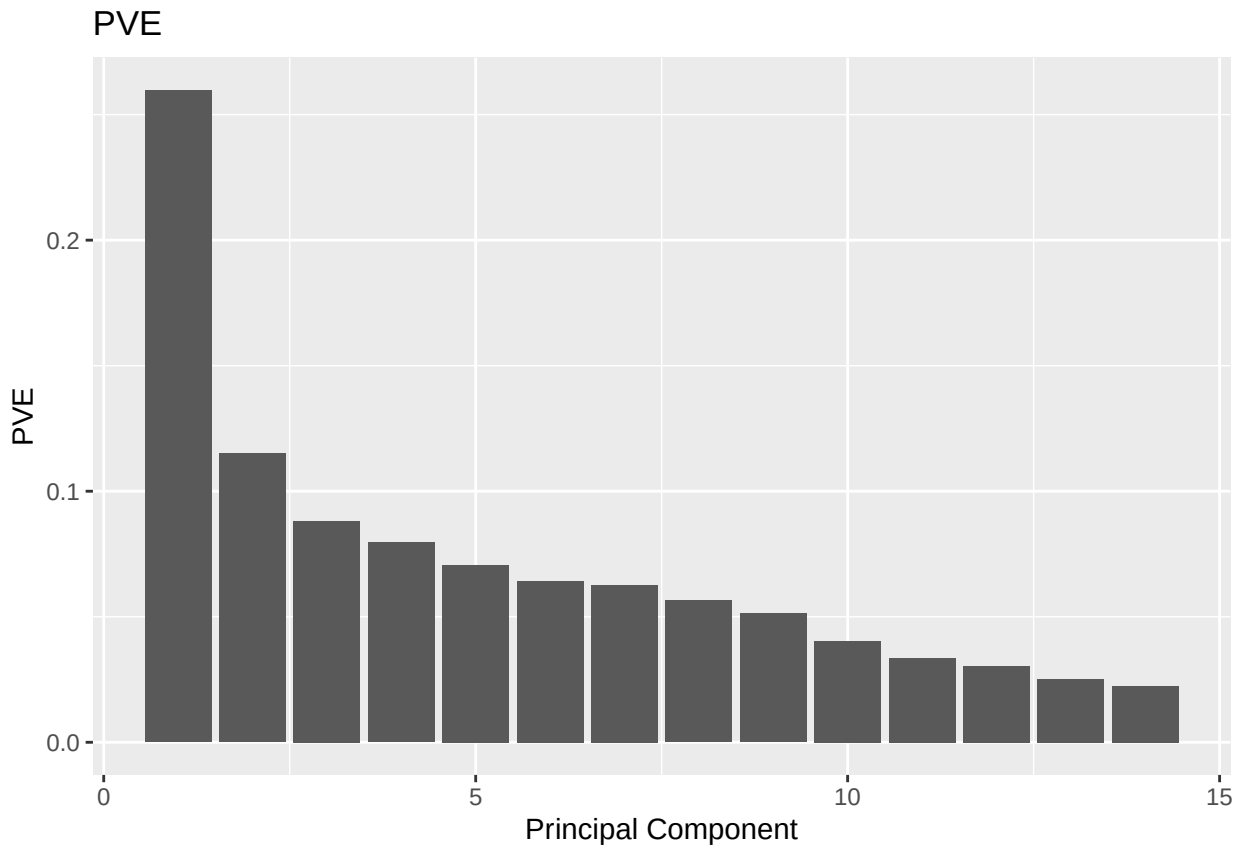
Plotten Sie die erste und zweite Hauptkomponente als Scatterplot. Färben Sie die Punkte gemäß ihrer Klasse (Disease) ein.

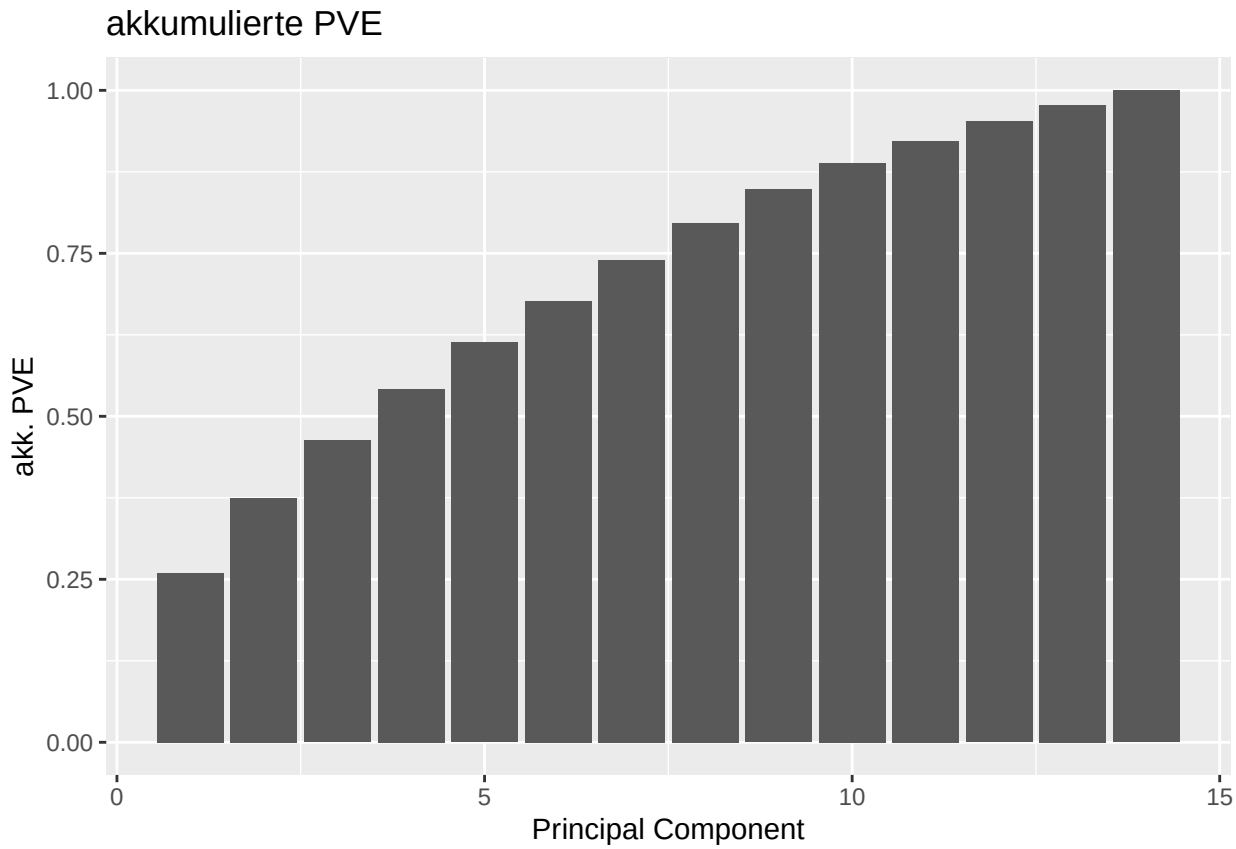
```
autoplot(pca_res, data=df_pca, color="goal")
```



PVE

Plot Plotten Sie die Proportion of Variance explained (PVE) für jede Hauptkomponente sowie die akkumulierte PVE.


PVE
PVE
0.2
0.1
0.0
0
5
10
15
Principal Component



Wichtige Hauptkomponenten Wie viele Hauptkomponenten erklären mehr als 50% der Varianz?

Möglicherweise tragen bei Ihrem Ergebnis die letzten Hauptkomponenten keine Varianz mehr bei. Überlegen Sie, woran das liegen könnte.

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.9067 1.2694 1.11034 1.0550 0.99422 0.94883 0.93717
## Proportion of Variance 0.2597 0.1151 0.08806 0.0795 0.07061 0.06431 0.06274
## Cumulative Proportion  0.2597 0.3748 0.46283 0.5423 0.61293 0.67724 0.73997
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.88959 0.84724 0.75262 0.68632 0.65265 0.59533 0.55976
## Proportion of Variance 0.05653 0.05127 0.04046 0.03365 0.03043 0.02532 0.02238
## Cumulative Proportion  0.79650 0.84777 0.88823 0.92188 0.95230 0.97762 1.00000
```

Hier erklären 4 Hauptkomponenten mehr als 50% der Varianz (siehe Zeile "Cumulative Proportion"). Die letzten Hauptkomponenten tragen noch zur Varianz bei, allerdings nur sehr wenig. In der PCA werden die Hauptkomponenten so angeordnet, dass die erste die höchste Varianz im Datensatz erklärt, die zweite die zweithöchste, usw. Die späteren Hauptkomponenten erklären sukzessive immer weniger Varianz im Vergleich zu den vorherigen.

Auf Datensätze bezogen gibt es oft Muster und Strukturen, die von einigen dominanten Merkmalen bzw. Hauptkomponenten gut erklärt werden können. Die letzten Hauptkomponenten repräsentieren feinere Details, die somit weniger Varianz erklären.